Memorias del 2022 Congreso Nacional de Control Automático
12-14 de Octubre, 2022. Tuxtla Gutiérrez, México.

Control Applications Using Reinforcement Learning: An Overview

Francisco Rodríguez Sánchez * **Ildeberto Santos-Ruiz** *
Joaquín Domínguez-Zenteno *
Francisco-Ronay López-Estrada *

* *Tecnológico Nacional de México, I.T. Tuxtla Gutiérrez, TURIX-Dynamics Diagnosis and Control Group, Carretera Panamericana km 1080 S/N, Tuxtla Gutiérrez 29050, Mexico; ildeberto.dr@tuxtla.tecnm.mx*

Abstract: This article presents the general formulation and terminology of reinforcement learning (RL) from the perspective of Bellman's equations based on a reward function, its learning methods and algorithms. The important key in RL is the calculation of value-state and value state-action functions, useful to find, compare and improve policies for learning agent through different methods based on values and policies such as Q-learning. The deep deterministic policy gradient (DDPG) learning algorithm based on an actor-critic structure is also described as one of the ways of training the RL agent. RL algorithms can be used to design closed loop controllers. Through simulation, using the DDPG algorithm, an example of the application of the inverted pendulum is proposed in simulation, demonstrating that the training is carried out in a reasonable time, showing the role and importance of RL algorithms, like tools that combined with control can address this type of problems.

Keywords: Reinforcement learning; Learning algorithm; DDPG; Control applications; Optimal control.

1. INTRODUCCIÓN

En los últimos años, el aprendizaje por refuerzo (*Reinforcement learning*, RL) ha recibido atención debido a la amplia gama de aplicaciones en problemas de toma de decisiones secuenciales a partir de la experiencia obtenida de un agente que interactúa con su entorno para optimizar objetivos en forma de recompensas acumulativas, lo cual impacta áreas como: sistemas de control robusto y óptimo (Recht, 2019), robots autónomos bípedos (García and Shafie, 2020), conducción autónoma (Kiran et al., 2021), planificación de tráfico urbano (Pan et al., 2021), entre otros. En sus inicios, el RL tuvo como principio el aprendizaje por ensayo y error; actualmente surge como una amalgama de control óptimo basado en costos, (Bertsekas, 2019) e inteligencia artificial basada en recompensas, (Sutton and Barto, 2018). Se han desarrollado algoritmos de entrenamiento del agente de RL, basado en funciones de valor, tales como: aprendizaje Q (*Q learning*), SARSA (*State, Action, Reward, State, Action*), red profunda Q (*Deep Q network*, DQN), entre otros. Q-learning fue propuesto por Watkins (1989) es considerado un algoritmo de aprendizaje discreto cuyo objetivo es llenar una tabla conocida como función Q en la que el agente obtendrá la mayor recompensa posible y la política sea óptima; está limitado a pequeños espacios de estado-acción. Una mejora del Q-learning es el algoritmo DQN sugerido por Mnih et al. (2013), es una combinación de Q-learning y redes convolucionales profundas (Wang et al., 2020); utilizadas como modelo para aproximar la política la cual depende de un parámetro. El algoritmo SARSA es otra mejora para el algoritmo Q-learning, basada en diferencias de tiempo para actualizar una función de valor, depende del estado actual (S), la acción actual (A), la recompensa obtenida (R), el siguiente estado (S') y la siguiente acción (A') simbolizada por la tupla (S, A, R, S', A'), dada una política para muestrear y evaluar su desempeño, (Gordon, 1996). Las aplicaciones típicas de RL con estos algoritmos incluyen, pero no se limitan, a: seguimiento de trayectoria (Tu et al., 2021), modelado del sistema de control motor humano (Song et al., 2021), entre otros. Sin embargo, aunque RL ha logrado varios resultados como lo demuestran las investigaciones citadas, queda mucho por hacer en diversas áreas de oportunidades, como: Bioelectrónica, tratamiento de los síntomas del Parkinson, (Avila et al., 2019), control de misiles (Li et al., 2020), locomoción de vehículos submarinos (Zhang et al., 2021), control

⋆ Esta investigación es financiada por el Tecnológico Nacional de México a través de la convocatoria Proyectos de Investigación Científica, Desarrollo Tecnológico e Innovación 2022. El trabajo de Francisco Rodríguez Sánchez también es financiado por CONACYT a través del Programa de Becas para Estudios de Posgrado.

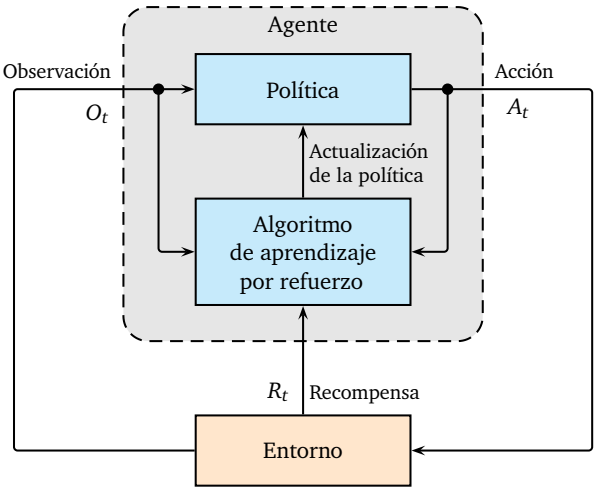

Figura 1. Esquema básico del aprendizaje por refuerzo.

magnético de plasma en recipientes TOKAMAK (Degrave et al., 2022). El documento presenta el RL como una herramienta que permite ser incluido como parte del diseño de estrategias de control. Los métodos del RL son de interés por las posibilidades teóricas y de sus aplicaciones potenciales. Presenta una visión general de las principales características de RL y su relación con el control óptimo. Este artículo está estructurado de la siguiente manera. La sección 2 está enfocado en los conceptos de RL, exponiendo los algoritmos para el cálculo de las funciones de valor de estado y de estado-acción desde el punto de vista de las ecuaciones de Bellman. El enfoque del control óptimo, es abordado en la sección 3. Le sigue la sección 4 donde se exponen los principales algoritmos de entrenamiento empleados por el agente de RL. En la sección 5 se presenta un ejemplo en simulación que muestra el uso práctico del algoritmo gradiente de política determinista profunda (*deep deterministic policy gradient*, DDPG). Finalmente, la sección 6 concluye el documento con comentarios útiles.

2. APRENDIZAJE POR REFUERZO

El RL es un proceso de toma de decisiones secuenciales que utiliza un conjunto de métodos para generar soluciones óptimas. Un sistema de RL está formado por el agente y el entorno interactuando en cada paso de tiempo (Figura 1) en una sucesión, $t = 0, 1, 2, \ldots$. En cada paso de tiempo, el agente interactúa con el entorno y recibe las observaciones O_t de los estados $s_t \in S$, donde S es el conjunto de estados; por la observación recibida, el agente emite una acción $A_t \in A(s_t)$, donde $A(s_t)$ es el conjunto de acciones disponibles en el estado s_t. Como consecuencia de sus acciones, el agente recibe una recompensa $R_t \in R$ y alcanza un nuevo estado s_{t+1}. (Tan et al., 2017). Comparado con otros paradigmas de aprendizaje automático, las características distintivas del RL son: basado en recompensas, los datos de entrenamiento son obtenidos de la interacción del agente con su entorno, las acciones se llevan a cabo mediante prueba y error. El enfoque de RL se basa en el concepto de recompensa R_t acumulada o retorno dado por (1), esta señal de retroalimentación del entorno al agente indica qué tan bien el agente en un estado particular S_t está haciendo la tarea al realizar una acción particular A_t; esta puede ser con o sin descuento. Las metas del agente se describen como la maximización de la expectativa de la recompensa acumulada o "retorno", dado por (2):

$$G_t = R_t + \gamma R_{t+1} + \cdots = \sum_{j=0}^{\infty} \gamma^j R_{t+j+1}, \tag{1}$$

$$\max_{\pi} \mathbb{E}\left[G_t\right], \tag{2}$$

donde $\gamma \in [0, 1]$ es el factor de descuento, G_t el retorno, $\mathbb{E}$ es la expectativa del retorno. La obtención de la política, implica calcular una función de valor, dada por (3); $V^u(s)$ es una función de valor de estado dada la política $u = \pi(s_j)$. La función de valor define la bondad acumulada por el agente a largo plazo por estar en un estado s dada una acción. La función de valor de acción $Q^u(s, a)$ dada la política $u = \pi(s_j)$, calcula qué tan bueno es el desempeño dada la acción a en el estado s (Liu et al., 2017), y se define como en (4):

$$V^u\left(s_0\right) = \sum_{j=0}^{\infty} \gamma^j r_{j+1}|_{s_0=s}, \tag{3}$$

$$Q^u\left(s, a\right) = \sum_{j=0}^{\infty} \gamma^j \, r_{t+j+1}|_{\{s_t=s, a_t=a\}}. \tag{4}$$

La política óptima denotada como π^* debe ser mejor o igual a una política conocida, de esta forma las funciones de valor de estado y de valor acción óptimo están definidas por (5) y (6). Bajo el supuesto que satisfacen el principio de optimización de Bellman (Liu et al., 2017),

$$V^*\left(s\right) = \arg\max_{a} \left\{r\left(s, a, s'\right) + \gamma V^*\left(s'\right)\right\}, \tag{5}$$

$$Q^*\left(s, a\right) = r\left(s, a, s'\right) + \gamma \max_{a'} \left\{Q^*\left(s', a'\right)\right\}. \tag{6}$$

La ecuacion (5) considera la recompensa inmediata más el cálculo futuro de la función de valor de la acción del siguiente estado. De manera semejante, la ecuación (6) considera la recompensa inmediata más la cantidad de recompensa que el agente espera recibir en el futuro. La política óptima se expresa reescribiendo (5) y (6) en la forma:

$$\pi^*\left(s\right) = \arg\max_{a} \left\{r\left(s, a, s'\right) + \gamma V^*\left(s'\right)\right\}, \tag{7}$$

$$\pi^*(s) = \arg\max_{a} \, Q^*(s, a). \tag{8}$$

Las ecuaciones (5) y (6) forman la base de los métodos de RL de iteración de políticas, iteración de valor y Q-learning. Algunos de los métodos populares de RL se exponen a continuación.

2.1 Métodos de aprendizaje por refuerzo

Calcular las funciones de valor de estado y de estado-acción es un paso muy importante de los métodos de RL, con el objetivo de mejorar la política, a continuación se mencionan algunos de estos métodos.

Evaluación de política El cálculo de la función de valor de estado $V^{\pi}(s)$ dada una política arbitraria π, se conoce como evaluación de política, (Sutton and Barto, 2018), esta se determina mediante la iteración descrita por (9):

$$V^{\pi}(s) = r(s, a, s') + \gamma V^{\pi}(s'), \tag{9}$$

comenzando en una aproximación inicial arbitraria V_0^{π}. Después de evaluar $V^{\pi}(s)$, se obtiene una política mejorada, dada por:

$$\pi'(s) = \arg\max_a \{r(s, a, s') + \gamma V^{\pi}(s')\}. \tag{10}$$

Q-learning Es un método de aprendizaje de diferencia temporal, (Tan et al., 2017), descrito por (11):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \tag{11}$$

Con base en la función de valor obtenida en (11), la política mejorada se determina en la forma (12):

$$\pi'(s) = \arg\max_a Q(s_t, a). \tag{12}$$

Iteración de valor, diferencia temporal, SARSA, entre otros, son métodos para obtener la función de valor, suelen nombrarse deterministas porque el cálculo de la política puede producir las mismas acciones en cualquier estado dado; contrario a los estocásticos, las acciones producidas están determinadas por la probabilidad de ocurrencia.

3. CONTROL ÓPTIMO MEDIANTE APRENDIZAJE POR REFUERZO

Se ha estudiado ampliamente el control óptimo en tiempo discreto o continuo, lineal o no lineal, invariante o variable en el tiempo; es un área en crecimiento, por lo que tiene una buena base teórica, (Lewis et al., 2012). Desde el punto de vista de la programación dinámica (*dynamic programming*, DP), el control óptimo se aproxima mediante la condición suficiente para resolver la ecuación de Hamilton-Jacobi-Bellman (HJB). Ambas herramientas DP y RL son utilizadas en la solución de problemas de decisión secuencial y muchos de los algoritmos de RL están fundamentados en la PD.

Caso discreto Se tiene un sistema no lineal en tiempo discreto definido como en (13):

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, \ldots, N-1, \tag{13}$$

donde, k es el índice de tiempo, $x_k \in R^n$ es el vector de estados, $u_k \in R^m$ es el vector de control, F es la función del sistema no lineal, N número de veces que se aplica el control. Comenzando en un estado inicial x_0 en R^n, es elegida una secuencia de control u_k, con $k = 0, 1, 2, \ldots$ tal que se logren ciertos objetivos cuando la secuencia de control se aplica al sistema (13), el siguiente índice de rendimiento (costo) está definido por (14):

$$J(x_k, \underline{u_k}) = \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i, u_i), \tag{14}$$

donde, $\underline{u_k} = (u_k, u_{k+1}, \ldots)$ es la secuencia de control que inicia en el tiempo k, $U(x_i, u_i) \geqslant 0$ es la función de utilidad, $\gamma \in [0, 1]$ es el factor de descuento. La función J depende del tiempo inicial k y del estado inicial x_k, y se conoce como el costo de atravesar el estado x_k. Se requiere determinar $\underline{u_0} = (u_0, u_1, \ldots)$ para que $J(x_0, \underline{u_0})$ sea óptimo. Dado $\underline{u_0^*} = (u_0^*, u_1^*, \ldots)$ y $J^*(x_0)$ para indicar la secuencia de control óptima y la función de costo óptima.

La acción de control, definida como $u_k = \mu(x_k), \forall k$, se conoce como política o ley de control. Para una política dada μ, la función de costo en (14) se reescribe como:

$$J^{\mu}(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i, \mu(x_i)), \tag{15}$$

la ecuación (15) es la función de costo del sistema (13), iniciando en x_k cuando la política $u_k = \mu(x_k)$ es aplicada. La función de costo óptima para el sistema (13) iniciando en x_0 es determinada como:

$$J^*(x_0) = \inf_{\mu} J^{\mu}(x_0) = J^{\mu^*}(x_0). \tag{16}$$

La acción de control óptima u_k^* en el tiempo k para que (16) alcance el mínimo está definida por:

$$u_k^* = \arg\min_{u_k} \{U(x_k, u_k) + \gamma J^*(x_{k+1})\}. \tag{17}$$

La ecuación (17) es el principio de optimalidad para sistemas en tiempo discreto.

Caso continuo Dado el siguiente sistema no lineal será definida la formulación en tiempo continuo:

$$\dot{x}(t) = F(x(t), u(t)), \quad t \geq t_0, \tag{18}$$

donde $x(t)$ es el vector de estados, $u(t)$ es el vector de control, y $F(x, u)$ representa la dinámica continua del sistema no lineal. El problema de control óptimo consiste en la optimización de una política de control calculada en términos de una función de costo dada una política μ. El objetivo es diseñar una acción de control $u(t) = \mu(x(t))$ para seguir un camino $x(t)$ que minimice el costo J, dado por (19):

$$J(x_0, u) = \int_{t_0}^{\infty} U(x(\tau), u(\tau))\, \mathrm{d}\tau, \tag{19}$$

donde $U(x, u) \geqslant 0$ es una función de utilidad y $x_0 = x(t_0)$. Por el principio de optimalidad de Bellman, el costo óptimo en los sistemas de tiempo continuo se define como:

$$J^*(x(t)) = \min_{u(t)} \{J(x(t), u(t))\}, \quad t \geq t_0. \tag{20}$$

La acción de control óptimo $u^*(t)$ será la que minimice la función de costo (19) y está dada por:

$$u^*(t) = \arg\min_{u(t)} \{J(x(t), u(t))\}, \quad t \geq t_0. \tag{21}$$

El tiempo de optimización es un factor muy importante en un sistema continuo; porque es un proceso iterativo la mayoría de las veces lento.

Tabla 1. Algoritmos de RL.

Agente	**Tipo**	**Espacio de acción**
Q-Learning	Basado en valor	Discreto
Red profunda Q (DQN)	Basado en valor	Discreto
SARSA	Basado en valor	Discreto
Gradiente de política (PG)	Basado en política	Discreto o continuo
Actor - Crítico (AC)	Actor - Crítico	Discreto o continuo
Optimización de políticas próximas (PPO)	Actor - Crítico	Discreto o continuo
Optimización de políticas de región de confianza	Actor - Crítico	Discreto o continuo
Gradiente de política determinista profunda (DDPG)	Actor - Crítico	Continuo
Gradiente de política determinista profunda con retraso doble (TD3)	Actor - Crítico	Continuo
Actor-Crítico suave (SAC)	Actor - Crítico	Continuo

4. ALGORITMOS DE APRENDIZAJE POR REFUERZO

En la Tabla 1 se enlistan algunos de los algoritmos para el entrenamiento del agente. Los algoritmos Q-Learning, SARSA y DQN son aptos para entornos con espacios de acción y observación discretos. En entornos con espacios de acción y observación continuo y basados en una estructura de red neuronal actor - crítico, DDPG es un algoritmo muy útil, TD3 es una extensión mejorada de DDPG. PPO requiere más tiempo de entrenamiento que DDPG. SAC genera políticas estocásticas. SARSA es un ejemplo de algoritmo, sobre la política (*on-policy*) calcula el valor de la política mientras la usa para el control.

4.1 Gradiente de política determinista profunda DDPG

El algoritmo DDPG funciona mediante el aprendizaje conjunto de la función Q óptima y la política, es fuera de la política (*off-policy*), es decir, determina la política óptima independientemente de las acciones del agente; es en línea. Con un enfoque actor-crítico, el agente se compone de cuatro redes neuronales, la función $Q(s,a|\theta^Q)$, la política $\mu(s|\theta^\mu)$ y sus respectivas copias o destinos $\theta^{Q'}$ y $\theta^{\mu'}$. La red $Q(s,a|\theta^Q)$ predice la recompensa acumulada para un estado s y una acción dada a. La política $\mu(s|\theta^\mu)$ mapea el estado con una acción. El ciclo de RL comienza con la inicialización aleatoria de las cuatro redes neuronales, (lineas I y II en el algoritmo). El agente actúa en el entorno para recolectar experiencias del buffer de dimensión D, (linea III), cada entrada del buffer es una tupla con la estructura (s_i, a_i, r_i, s_{i+1}), donde s_{i+1} representa el nuevo estado después de tomar la acción a. Para aproximar la función Q óptima, el error cuadrático medio se minimiza empleando la función de pérdida dada en el punto (6); donde $Q\left(s_i, a_i|\,\theta^Q\right)$ es la red neuronal con la aproximación actual de la función Q óptima, N es el número de transiciones aleatoriamente del buffer de experiencia, s_i es la observación del estado siguiente y y_i es la función de valor objetivo definida por el punto (5). A partir de la actualización de $Q(s,a|\theta^Q)$, se calcula el gradiente punto (7) para actualizar la política $\mu(s|\theta^\mu)$.

Algoritmo DDPG *Basado en Lillicrap et al. (2015)*

Usado para actualizar el modelo neuronal actor-crítico en cada paso de tiempo.

Init:

I. Inicializar aleatoriamente la red crítico $Q(s,a|\theta^Q)$ y la red actor $\mu(s|\theta^\mu)$ con pesos θ^Q y θ^μ.
II. Inicializar la red objetivo Q' y μ' con pesos: $\theta^{Q'} \leftarrow \theta^Q, \qquad \theta^{\mu'} \leftarrow \theta^\mu$
III. Inicializar el buffer de repetición D, con las muestras de las transiciones (s_i, a_i, r_i, s_{i+1}).

for episodio **from** 1 **to** M **do**

a. Inicializar un proceso aleatorio $\mathcal{N}$ para la exploración de las acciones.
b. Recibir el estado de observación inicial s_1.
c. **for** t **from** 1 **to** T **do**

1. Seleccionar la acción $a_i = \mu\left(s_i|\theta^\mu\right) + \mathcal{N}_i$ de acuerdo con la política actual y el ruido de exploración.
2. Ejecutar la acción a_i y observar la recompensa r_i y el nuevo estado s_{i+1}.
3. Almacenar las transiciones (s_i, a_i, r_i, s_{i+1}) en D.
4. Muestrear aleatoriamente un minilote de N transiciones (s_i, a_i, r_i, s_{i+1}) de D.
5. Establecer:
$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)|\,\theta^{Q'}\right)$$
6. Actualizar la red crítico minimizando la función de pérdida:
$$L = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q\left(s_i, a_i|\,\theta^Q\right)\right)^2$$
7. Actualizar la política de la red actor utilizando el gradiente de política muestreado:
$$\nabla_{\theta^\mu} J = \frac{1}{N}\sum_{i=1}^{N} \nabla_a Q\left(s, a|\theta^Q\right)\Big|_{\substack{s=s_i \\ a=\mu(s_i)}} \nabla_{\theta^\mu}\mu\left(s|\theta^\mu\right)|_{s_i}$$
8. Actualizar las redes objetivo:
$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\,\theta^{Q'}, \quad \theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\,\theta^{\mu'}$$

end for

end for

Finalmente, los parámetros de las redes neuronales objetivo se actualiza, punto (8), usando un descenso de gradiente estocástico.

5. EJEMPLO DE APLICACIÓN: ESTABILIZACIÓN DE UN PÉNDULO INVERTIDO

A continuación se describe una simulación en MATLAB-Simulink que usa un agente DDPG para estabilizar un péndulo invertido.

El sistema, consiste en una barra rígida montada sobre un carro impulsado por una fuerza (producida por un motor) que lo desplaza horizontalmente sobre un riel. La barra gira libremente sobre su apoyo que le sirve como eje de rotación (se desprecia la fricción angular y la fricción de Coulomb del carro), como se muestra en la Figura 2.

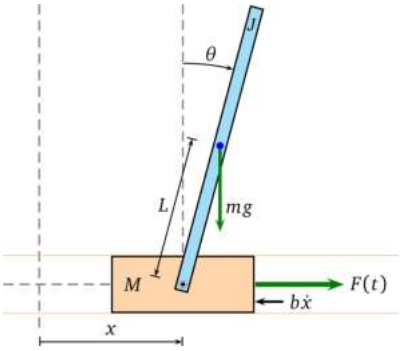

Figura 2. Sistema carro-péndulo.

Tabla 2. Parámetros del carro-péndulo.

Símbolo	Valor	Descripción
M	0.5 kg	Masa del carro
m	0.2 kg	Masa de la barra
L	0.6 m	Distancia al centro de masa
J	0.006 kg m^2	Momento de inercia del péndulo
g	9.8 m/s^2	Aceleración de la gravedad
b_x	0.1 Ns/m	Coeficiente de fricción viscosa

La ecuación (22) representa el modelo lineal del carro - péndulo, con los parámetros mostrados en la Tabla 2. Los estados que describen la dinámica del sistema son: $x_1 = x$ posición lineal del carro, $x_2 = \dot{x}$ velocidad lineal del carro; con respecto a la vertical, $x_3 = \theta$ posición angular de la barra, $x_4 = \dot{\theta}$ velocidad angular de la barra,

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -(J + mL^2 b_x)/\Delta & m^2L^2g/\Delta & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -mLb_x/\Delta & (m+M)gmL/\Delta & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ (J + mL^2)/\Delta \\ 0 \\ mL/\Delta \end{bmatrix} u(t), \quad (22)$$

donde $\Delta = J(m+M)+MmL^2$. La acción, las observaciones y la recompensa están definidas de la siguiente forma:

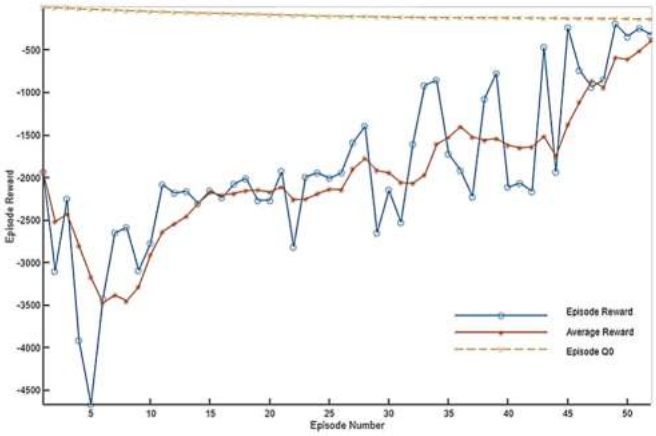

Figura 3. Entrenamiento del agente DDPG.

La **acción** del agente al entorno es $-15\,\text{N} \leq u(t) \leq 15\,\text{N}$. Las **observaciones** del entorno son los estados del sistema: x, $\dot{x}$, θ y $\dot{\theta}$. La función de **recompensa** r_t, que depende de los estados y de la acción, se definió por:

$$r_t = -0.1 \times (5\theta^2 + x^2 + 0.05u^2) - 100B, \quad (23)$$

donde $\theta_{\text{máx}} \in [-10°, 10°]$, $x \in [-1.5\,\text{m}, 1.5\,\text{m}]$, y B es una bandera binaria que se activa cuando el carro alcanza sus límites de recorrido máximo. Cada episodio del entrenamiento termina cuando el desplazamiento del carro es tan grande que excede un umbral de distancia respecto al origen.

El progreso del entrenamiento del agente DDPG presentado en la Figura 3, muestra la recompensa de cada episodio (EpisodeReward) y la recompensa promedio (AverageReward); en la gráfica se observa que, el EpisodeQ_0 a medida que avanza el entrenamiento, el agente elige los pares de estado-acción que le generen mayor recompensa. Los parámetros de entrenamiento utilizados se muestran en la Tabla 3. Para el entrenamiento se estableció un máximo de 2000 episodios, el número máximo de pasos por episodio fue de 1250. El entrenamiento se detuvo cuando la recompensa promedio de cinco episodios consecutivos llegó a -400.

Tabla 3. Parámetros de entrenamiento.

Descripción	Valor
Tasa de aprendizaje de la red Actor	5×10^{-4}
Tasa de aprendizaje de la red Crítico	1×10^{-3}
Umbral del gradiente	1.0
Longitud del búfer de experiencia	1×10^{6}
Longitud del búfer de repetición	128
Periodo de muestreo	0.02 s
Tiempo de simulación	0.99 s

Al finalizar el entrenamiento, el sistema carro-péndulo se estabilizó como se muestra en la Figura 4, alcanzando una posición del carro alrededor del punto $x = 0$, lo que equivale la mitad del riel. Durante el entrenamiento, la velocidad lineal del carro $\dot{x}$ presentó variaciones grandes

en algunos puntos, quedando estable alrededor de 0.2 m/s al finalizar el entrenamiento. El ángulo θ, que es el objetivo del entrenamiento, quedó estable alrededor de 6.3°, por debajo de los 10° establecidos como máximo; en este punto se alcanzó una velocidad angular $\dot{\theta}$ prácticamente de cero.

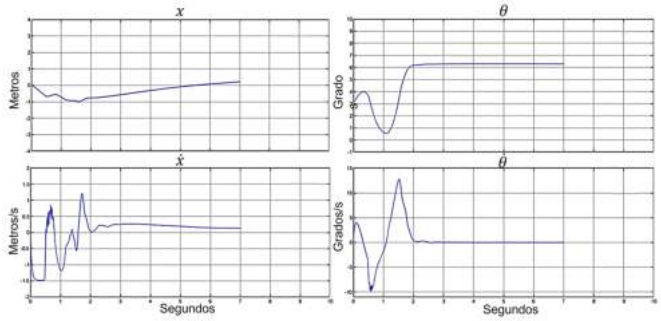

Figura 4. Comportamiento de los estados del carro - péndulo.

6. CONCLUSIÓN

El RL se ha aplicado con éxito en conducción autónoma de vehículos, robótica y videojuegos, entre otros usos, siendo de especial interés las aplicaciones al control. Este artículo se centró en presentar el RL desde el punto de vista de la ecuación de Bellman, con una descripción de los diferentes métodos para calcular la función de valor, como la evaluación de políticas. El enfoque del RL también se describió desde el punto de vista del control óptimo, discreto y continuo. Se presentó una simulación para ejemplificar el comportamiento del agente DDPG, mostrando que el aprendizaje por refuerzo amplía las posibilidades de solución basadas en datos para problemas de optimización y control.

REFERENCIAS

Avila, L., De Paula, M., Carlucho, I., and Reinoso, C.S. (2019). Mppt for pv systems using deep reinforcement learning algorithms. *IEEE Latin America Transactions*, 17(12), 2020–2027.

Bertsekas, D. (2019). *Reinforcement learning and optimal control*. Athena Scientific.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897), 414–419.

García, J. and Shafie, D. (2020). Teaching a humanoid robot to walk faster through safe reinforcement learning. *Engineering Applications of Artificial Intelligence*, 88, 103360.

Gordon, G.J. (1996). Chattering in sarsa (λ). *CMU Learning Lab Technical Report*.

Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A.A., Yogamani, S., and Pérez, P. (2021). Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*.

Lewis, F.L., Vrabie, D., and Syrmos, V.L. (2012). *Optimal control*. John Wiley & Sons.

Li, B., Ma, F., and Wu, Y. (2020). Missile attitude control based on deep reinforcement learning. In *2020 IEEE 16th International Conference on Control & Automation (ICCA)*, 931–936. IEEE.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Liu, D., Wei, Q., Wang, D., Yang, X., and Li, H. (2017). *Adaptive dynamic programming with applications in optimal control*. Springer.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Pan, T., Guo, R., Lam, W.H., Zhong, R., Wang, W., and He, B. (2021). Integrated optimal control strategies for freeway traffic mixed with connected automated vehicles: A model-based reinforcement learning approach. *Transportation research part C: emerging technologies*, 123, 102987.

Recht, B. (2019). A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2, 253–279.

Song, S., Kidziński, Ł., Peng, X.B., Ong, C., Hicks, J., Levine, S., Atkeson, C.G., and Delp, S.L. (2021). Deep reinforcement learning for modeling human locomotion control in neuromechanical simulation. *Journal of neuroengineering and rehabilitation*, 18(1), 1–17.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tan, F., Yan, P., and Guan, X. (2017). Deep reinforcement learning: From q-learning to deep q-learning. In *International Conference on Neural Information Processing*, 475–483. Springer.

Tu, Y., Fang, H., Yin, Y., and He, S. (2021). Reinforcement learning-based nonlinear tracking control system design via ldi approach with application to trolley system. *Neural Computing and Applications*, 1–8.

Wang, Z.T., Ashida, Y., and Ueda, M. (2020). Deep reinforcement learning control of quantum cartpoles. *Physical Review Letters*, 125(10), 100401.

Watkins, C.J.C.H. (1989). Learning from delayed rewards. *PhD thesis, Cambridge University*.

Zhang, T., Wang, R., Wang, Y., and Wang, S. (2021). Locomotion control of a hybrid propulsion biomimetic underwater vehicle via deep reinforcement learning. In *2021 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 211–216. IEEE.